Preprint: Software Quality, Reliability and Security Companion (QRS-C), 2017 IEEE International Conference on, pp. 61-68. IEEE, 2017

Combinatorial and MC/DC Coverage Levels of Random Testing

Sergiy Vilkomir, Aparna Alluri
Department of Computer Science
East Carolina University
Greenville, NC 27858, USA
vilkomirs@ecu.edu

D. Richard Kuhn, Raghu N. Kacker
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899, USA
{kuhn, raghu.kacker}@nist.gov

***Abstract*—Software testing criteria differ in effectiveness, numbers of required test cases, and a process of test generation. Specific criteria are often compared with random testing as a simplest basic approach and, in some cases, random testing shows a surprisingly high level of effectiveness. One of the reasons is that any random test set has a specific level of coverage according to any coverage criterion. Numerical evaluation of coverage levels of random testing according various coverage criteria is interesting research task important for understanding relationship between different testing approaches. In this paper we experimentally evaluate coverage levels of random testing for two criteria – MC/DC and combinatorial *t*-way testing. The results could be used for selection optimal methods for practical testing, and development of new testing methods based on integration of existing approaches.**

Keywords—random testing; combinatorial testing; MC/DC; pairwise; coverage

I. INTRODUCTION

Testing coverage criteria are widely used in software testing. According to ISO/IEC/IEEE 29119-1:2013 Standard [1], test coverage is "a degree, expressed as a percentage, to which specified test coverage items have been exercised by a test case or test cases". Simple examples include statement coverage, path coverage, and branch coverage. A more sophisticated example is *t*-way combinatorial coverage, which requires every possible combination of values of *t* parameters be included in some test case in the test suite [2, 3]. In the most basic form for *t*=2, this criterion is known as pairwise testing and requires all possible pairs of values be covered [4]. One of the strongest coverage criteria for testing logical predicates is Modified Condition/Decision Coverage (MC/DC) [5], which requires every condition in a decision be covered, and "each condition has been shown to independently affect the decision's outcome" [6]. MCDC coverage subsumes branch coverage, and in turn, statement coverage of code.

Because MCDC is such a strong criterion, the US Federal Aviation Administration (FAA) has for many years required MCDC testing for Level A (life critical) software aboard commercial aircraft [7, 8], but it is rarely used outside of the aerospace industry. One of the most significant barriers to wider use of MCDC is its expense. While testing for consumer grade software is roughly the same as the cost of producing the code, in the aviation industry spending could be seven times more on verification than development [9]. To encourage wider use of MCDC beyond the aerospace industry, the cost of its application will need to be reduced.

Testing criteria differ in effectiveness, numbers of required test cases, and a process of test generation. Empirical evaluation and experimental comparison of testing criteria started in 1980's [10, 11] but this is still an important research direction [12, 13, 14]. Specific criteria are often compared with random testing as a simplest basic approach [15, 16, 17, 18].

In some cases, random testing shows a surprisingly high level of effectiveness. For example, for testing logical expressions, *t*-way testing have an advantage over random testing but this benefit is not significant [18, 19]. One of the reasons is that any random (or any other) test set has a specific level of coverage according to any coverage criterion. Thus, random testing has a certain level of pairwise coverage, etc. This level is never 100% but could be quite high. Numerical evaluation of coverage levels of random testing according various coverage criteria is interesting and important research task. It could be useful for understanding relationship between different testing approaches, selection optimal methods for practical testing, and development of new testing methods based on integration of existing approaches.

In this paper we experimentally evaluate coverage levels of random testing for two criteria – MC/DC and *t*-way testing. The paper is structured as follows: Section II considers a general problem of the integration of testing techniques to reduce the number of test cases and increase the code coverage and testing effectiveness. As a part of this problem, the more specific research question about evaluation of the MC/DC and combinatorial coverage levels of random testing is discussed. The methods and scope of the investigation are described in Section III. Section IV provides experimental results on the MC/DC coverage level of random testing and Section V provides similar results for combinatorial coverage levels. Conclusions and directions for future work are presented in Section VI.

II. RESEARCH QUESTION

Testing according to any coverage criterion, as well as other testing approaches, has some benefits and some challenges. For example, MC/DC has good effectiveness for testing logical expressions but the process of test generation to satisfy MC/DC is quite complicated. Random testing could be effective but usually only for a large number of test cases.

Combinatorial criteria are very effective in many situations but not for testing logical expressions.

Taking this into consideration, a natural idea is trying to combine different testing approaches to use advantages of each of them. The purpose of such combination is to minimize a number of test cases, maximize effectiveness of testing, and simplify test generation as much as possible. Researches in this direction include combining functional and structural testing [20, 21], model-based and combinatorial testing [22], model-based and search-based testing [23] and more.

Some preliminary research suggests that using combinatorial testing in conjunction with model-based approaches can significantly reduce the cost of achieving MCDC coverage [24]. Part of the savings occurs because tests based on *t*-way covering arrays will necessarily also cover some proportion of MCDC and branch coverage. For example, a test set covering all *2*-way combinations of binary variable settings will ensure that branch predicates containing only two binary variables will be instantiated in all possible ways. But *2*-way (pairwise) test sets will also cover some proportion of *t*-way combinations for all $t > 2$, up to n, where n is the number of variables. Similarly, tests with random values will also cover a significant proportion of *t*-way combinations.

Even when two criteria are completely independent and use different principles, a test set satisfied the first criterion has some level of coverage according to the second criterion and opposite. For example, 100% MC/DC test set has some pairwise coverage and the test set provided 100% pairwise coverage also provides a certain percentage of MC/DC coverage. This fact is well-known and there are some initial results on such relationships (i.e., for pairwise and MC/DC testing [14]) but this area still requires more numerical evaluations based on empirical investigations.

Because of its simplicity, random testing is a good candidate for combining with coverage criteria. For this purpose it is necessary to understand how good random testing is in providing coverage for different criteria and how level of coverage changes depending on the number of test cases in the random test set. We consider two specific research questions:

- RQ1: What is the level of MC/DC coverage of random test sets of different sizes?
- RQ2: What is the level of *t*-way coverage of random test sets of different sizes?

Answers on these questions do not solve the problem of combining different testing criteria but are necessary and important steps in this direction.

III. Methods and Scope of Investigation

A. Used tools

To measure MC/DC coverage, we used CodeCover and Testwell CTC++ tools. CodeCover is an open-source white-box testing tool developed at the University of Stuttgart, Germany in 2007 [25, 26]. CodeCover measures several types of coverage including term coverage (subsumes MC/DC) and supports several programming languages including Java and C. Testwell CTC++ is a code coverage and dynamic analysis tool for C and C++ code but also supports Java, and C#. The tool has been developed by Testwell Ltd company (Finland) [27] and since 2013 is owned by Verifysoft Technology GmbH [28]. As well as CodeCover, CTC++ provides measurements of several types of coverage including MC/DC. A sample report produced by Testwell CTC++ is presented in Fig. 1.

We used two tools to evaluate MC/DC levels because results of such evaluation for the same software and the same test sets often are different for different tools. The main reason for this is that there is no commonly accepted definition of an incomplete (not 100%) MC/DC coverage. Different principles are used in different tools. Some tools evaluate coverage separately for each logical condition and calculate an average value then. Other tools create complete MC/DC test sets, compare them with actually used test sets, and evaluate percentage of coverage based on this.

```
HITS/TRUE      FALSE    LINE DESCRIPTION
=============================================================================

        2                6 FUNCTION main()
        1          1    10   if (myfile . is_open ( ))
        8          1    12     while (getline ( myfile , line ))
       24          8    19       while (iss >> val)
                        22       }+
        3          5    26       if (( a && b ) || ( a && c ))
        2               26         1: (T && T) || (_ && _)
        1               26         2: (T && F) || (T && T)
        0               26         3: (F && _) || (T && T)
                   1    26         4: (T && F) || (T && F)
                   0    26         5: (T && F) || (F && _)
                   0    26         6: (F && _) || (T && F)
                   4    26         7: (F && _) || (F && _)
                        26         MC/DC (cond 1): 1 + 7, 1 - 6
                        26         MC/DC (cond 2): 1 + 4, 1 - 5
                     -  26         MC/DC (cond 3): 2 - 5, 3 - 7
                        26         MC/DC (cond 4): 2 + 4, 3 - 6
                        26       }+
                        27       else
                        28       }+
                        30     }+
                        31     else
                        31     }+
        2               32     return 0
                        33 }

***TER  93 % ( 13/ 14) of FUNCTION main()
      100 % ( 21/ 21) statement
```

Fig. 1. Testwell CTC++ sample report.

It is not our task in this paper to judge different approaches to MC/DC evaluation. There are some justifications for all of them. However, we provide the results of MC/DC evaluation from two tools and compare them in Section 4.

To measure *t*-way coverage, we used the Combinatorial Coverage Measurement (CCM) tool developed by National Institute of Standards and Technology (NIST) and the Centro Nacional de Metrologia of Mexico [29, 30]. CCM can analyze existing levels of *t*-way (t=2…6) coverage for any test set. Fig. 2 shows CCM tool user interface. The tool displays a graph showing the coverage for the given tests. Additional tests can be added to increase the coverage. The percentage of coverage can be viewed in the "Results" tab.

B. Main steps and the scope of the investigation

The general organization of experimental evaluation is illustrated in Fig. 3.

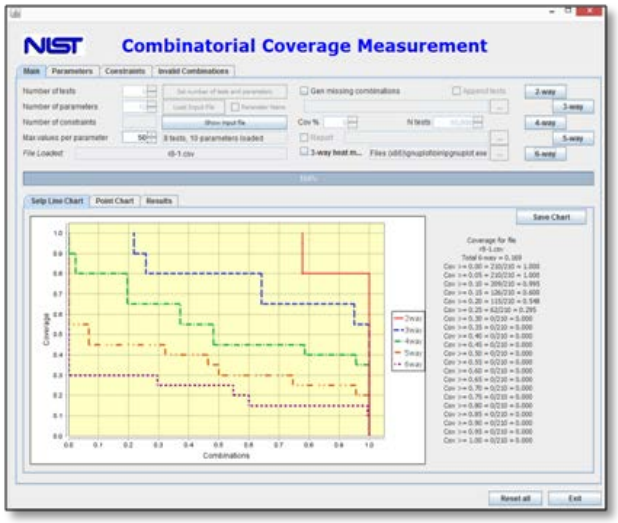

Fig. 2. CCM tool user interface.

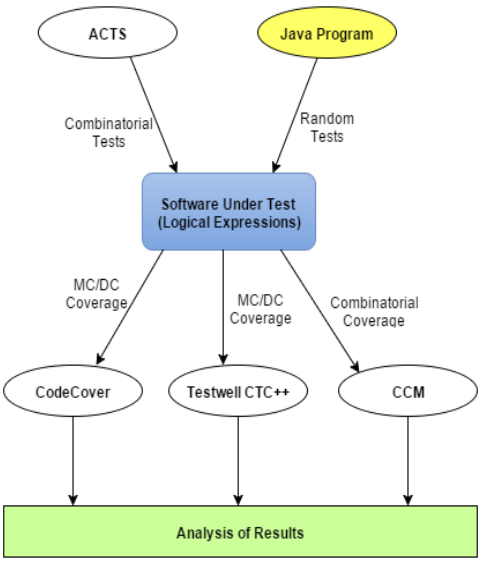

Fig. 3. Organization of experimental evaluation.

The investigation included the following main steps:

Step 1. Generation of logical expressions of different sizes (i.e., different numbers of logical variables in expressions). The sets of expressions were put into software programs which are without real functionality but used only for testing purposes to evaluate coverage levels. A total of 100 logical expressions were generated for testing. 50 expressions were generated from 10 fixed input variables and another 50 from 20 fixed input variables. Both sets contain 25 simple and 25 complex expressions of different sizes: 25 expressions of size 3, 15 of size 4, 5 of size 5, 2 of size 6, and 1 of each size 7, 8, and 9. Some examples of the generated expressions from 10 variables are presented in Table I. The sizes of expressions were chosen in such a way that they reflect situations in real software i.e. more expressions of small size and less of large size. The used proportion of different sizes approximately corresponds with data on expression sizes reported at [31, 32].

TABLE I. EXAMPLES OF LOGICAL EXPRESSIONS

	Simple expressions	**Complex expressions**
Size 3	d∨f∨c	(i∨!d) ∧ (!i∨!e)
Size 4	(c∨f) ∧ (i∨g)	(f∧ (a∨h)) ∨ (!h∧e∨!f)
Size 5	(c∧e) ∨ (!a∧f∨d)	(!h∨!f) ∧ (!c∨h∨e∧a) ∧ (f∨!e)

Step 2. Generation of random test sets of different sizes, where the size of a random test set is the number of test cases in it. To generate test cases and check that all test cases in a test set are different, we create a simple software program which used Java random number generator. 42 random test sets of different sizes were generated for 10 variables and 42 similar sets for 20 variables. The sizes of the generated random test sets were 1, 2, 3,… 25, 30, 40,…100, 200,….900, 1024. For each size, 3 test sets were generated, so the coverage for any random test size was the average of the three coverage values.

Step 3. Evaluation of MC/DC coverage for random test sets using the CodeCover tool.

Step 4. Evaluation of MC/DC coverage for random test sets using the Testwell CTC++ tool.

Step 5. Evaluation of *t*-way coverage for random test sets using the CCM tool.

Step 6. Analysis of experimental data.

The total numbers of logical expressions and test sets are summarized in Table II.

TABLE II. SCOPE OF EXPERIMENTAL TESTING

Number of		
Logical Variables	**Logical Expressions of mixed sizes**	**Random Test Sets**
10	50	126
20	50	126
Total	100	252

IV. MC/DC LEVEL OF RANDOM TESTING

All test inputs in generated random test sets have values T (True) or F (False). The example of the test set of size 5 as one of 42 generated random test sets is presented in Table III. Detailed data on MC/DC coverage by CoderCover and CTC++ tools for all random test sets are given in Table IV.

TABLE III. EXAMPLE OF THE RANDOM TEST SET OF SIZE 5

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**
1	T	T	F	F	T	F	F	F	T	F
2	T	F	T	F	F	T	T	F	F	F
3	T	F	T	F	T	T	T	T	T	F
4	T	T	F	F	F	T	F	F	F	F
5	F	T	F	F	F	T	T	T	T	F

The obtained results showed that MC/DC coverage demonstrated a fast growing trend when the number of random test cases increased. This trend was shown both by CoderCover (Fig. 4) and CTC++ (Fig. 5) tools for the simple

and complex expressions. For the simple expressions, MC/DC coverage reached 99% level for 55 random test cases and complete 100% MC/DC coverage was achieved started from 100 tests. The results by CoderCover and CTC++ for simple expressions were similar and close to each other (Fig. 6).

For complex expressions, more test cases were required to reach maximum MC/DC coverage. The coverage was very close to maximum from 200 test cases and maximum MC/DC coverage required approximately 400 random test cases. However, it is necessary to mention two distinctive features of estimation of MC/DC coverage for complex expressions:

- The MC/DC levels reported by two tools were significantly different (Fig. 7).
- The maximum of MC/DC coverage did not reach 100%. Even for exhaustive testing (all 1024 possible test cases for 10 variables), the maximum level of MC/DC coverage was 93% according to CodeCover and only 77% according to CTC++.

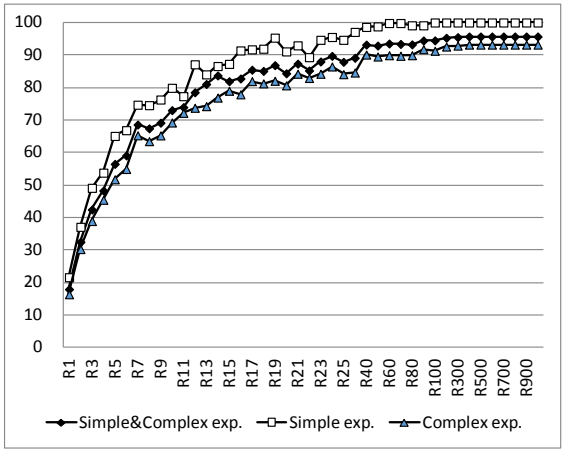

Fig. 4. MC/DC coverage by CodeCover for Random tests (10 variables).

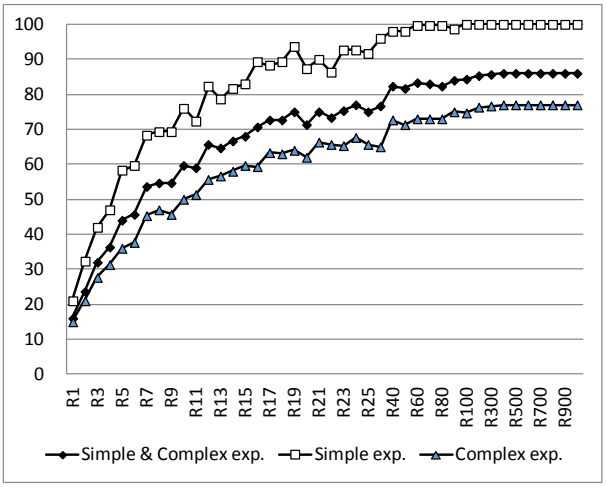

Fig. 5. MC/DC coverage by Testwell CTC++ for Random tests (10 variables.).

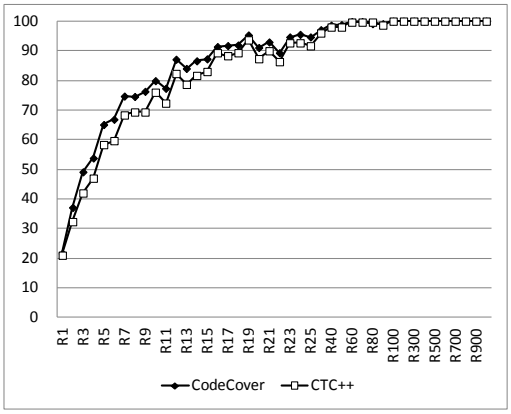

Fig. 6. Comparison of CodeCover and Testwell CTC++ results for Simple Expressions (10 variables).

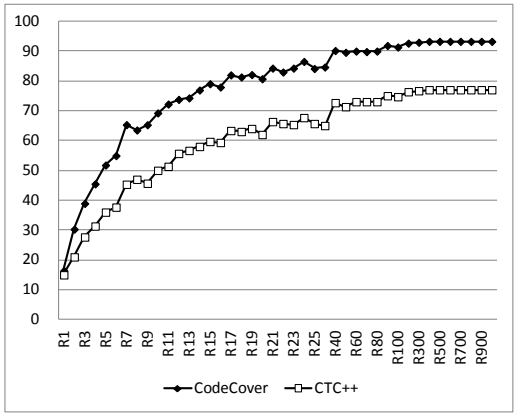

Fig. 7. Comparison of CodeCover and Testwell CTC++ results for Complex Expressions (10 variables).

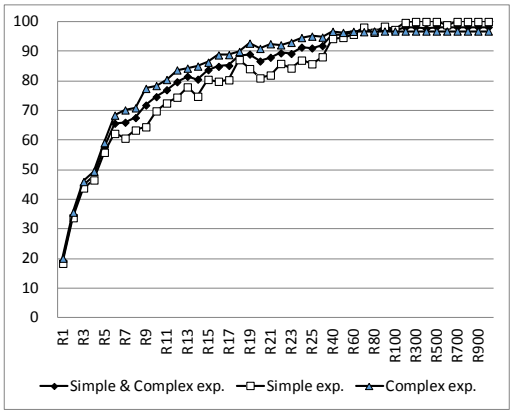

Fig. 8. MC/DC coverage by CodeCover for Random tests (20 variables).

The similar situation is not only for CodeCover and CTC++ but also for other tools provided MC/DC coverage, for example, Kalimetrix Logiscope [33] or TESSY [34]. The reason, as it was mentioned in Section 3.A, could be that different tools could use different principles for coverage evaluation. They also can evaluate different types of MC/DC, for example, Unique-Cause MC/DC vs. Masking MC/DC.

Other factors that affect evaluation of coverage are how tools treat short-circuit Boolean expressions and multiple occurrences of logical variables in expressions. Thus, in our case, CTC++ considers multiple occurrences of the same variable as different variables and requires test cases with different values of the first and second occurrences of the variable, that is impossible. Under this definition used, complex expressions never can have 100% MC/DC coverage. In contrast with CTC++, CodeCover considers multiple occurrences as the same variable so 100% MC/DC coverage for complex expressions is possible. However, when the short-circuit operator is used for evaluation of expression values, CodeCover considers some variables as uncovered even if test cases provide necessary coverage.

TABLE IV. MC/DC COVERAGE FOR RANDOM TESTS (10 VARIABLES)

Random Test Set Size	MC/DC Coverage for Simple and Complex Expressions		MC/DC Coverage for Simple Expressions		MC/DC Coverage for Complex Expressions	
	CodeCover	Testwell CTC++	CodeCover	Testwell CTC++	CodeCover	Testwell CTC++
1	17.97	16.00	21.63	21.00	16.40	15.00
2	32.50	23.67	37.13	32.33	30.30	21.00
3	42.47	32.00	49.13	42.00	38.93	27.67
4	48.30	36.33	53.80	47.00	45.47	31.33
5	56.50	44.00	65.10	58.33	51.77	36.00
6	59.10	45.67	66.87	59.67	54.93	37.67
7	68.60	53.67	74.70	68.33	65.30	45.33
8	67.43	54.67	74.57	69.33	63.50	47.00
9	69.17	54.67	76.30	69.33	65.27	45.67
10	73.03	59.67	79.90	76.00	69.23	50.00
11	73.97	59.00	77.33	72.33	72.20	51.33
12	78.53	65.67	87.13	82.33	73.77	55.67
13	81.07	64.67	84.03	78.67	74.27	56.67
14	83.70	66.67	86.60	81.67	76.90	58.00
15	81.93	68.00	87.27	83.00	79.00	59.67
16	82.77	70.67	91.40	89.33	77.90	59.33
17	85.47	72.67	91.73	88.33	81.97	63.33
18	85.10	72.67	91.93	89.33	81.27	63.00
19	86.87	75.00	95.33	93.67	82.07	64.00
20	84.37	71.33	91.10	87.33	80.67	62.00
21	87.33	75.00	92.97	90.00	84.23	66.33
22	85.23	73.33	89.33	86.33	82.93	65.67
23	88.00	75.33	94.67	92.67	84.23	65.33
24	89.73	77.00	95.53	92.67	86.47	67.67
25	87.90	75.00	94.67	91.67	84.10	65.67
30	89.13	76.67	97.10	96.00	84.60	65.00
40	93.20	82.33	98.63	98.00	90.13	72.67
50	92.90	81.67	98.80	98.00	89.53	71.33
60	93.53	83.33	99.83	99.67	89.93	73.00
70	93.40	83.00	99.83	99.67	89.77	73.00
80	93.27	82.33	99.13	99.67	89.93	73.00
90	94.47	84.00	99.13	98.67	91.80	75.00
100	94.50	84.33	100.00	100.00	91.30	74.67
200	95.30	85.33	100.00	100.00	92.60	76.33
300	95.50	85.67	100.00	100.00	92.90	76.67
400	95.70	86.00	100.00	100.00	93.20	77.00
500	95.70	86.00	100.00	100.00	93.20	77.00
600	95.70	86.00	100.00	100.00	93.20	77.00
700	95.70	86.00	100.00	100.00	93.20	77.00
800	95.70	86.00	100.00	100.00	93.20	77.00
900	95.70	86.00	100.00	100.00	93.20	77.00
1024	95.70	86.00	100.00	100.00	93.20	77.00

To investigate how the total number of logical variables in software affects MC/DC coverage of random testing, we repeated testing with 20 variables instead of 10 variables. The sizes of logical expressions remained the same (from 3 to 9) but variables for each expression were selected from the set of 20 variables. Each random test case had also 20 input values though not all of them where used for each expression. Detailed data for this testing are presented in Table V and Fig. 8 and 9.

Some conclusions were similar for 10 and 20 variables:

- The levels of MC/DC coverage for simple expressions by CodeCover and CTC++ were close each other (Fig.10).
- The levels of MC/DC coverage for complex expressions by CodeCover and CTC++ were significantly different (Fig. 11) and CodeCover reported higher levels of coverage.

TABLE V. MC/DC COVERAGE FOR RANDOM TESTS (20 VARIABLES)

Random Test Set Size	MC/DC Coverage for Simple and Complex Expressions		MC/DC Coverage for Simple Expressions		MC/DC Coverage for Complex Expressions	
	CodeCover	Testwell CTC++	CodeCover	Testwell CTC++	CodeCover	Testwell CTC++
1	19.10	16.00	18.40	20.00	20.13	17.00
2	34.60	26.67	33.80	31.67	35.67	26.00
3	44.87	34.00	43.83	38.67	46.00	33.33
4	48.03	39.00	46.53	43.33	49.50	37.33
5	57.57	43.67	55.87	49.00	59.13	42.33
6	65.67	51.00	62.23	53.67	68.40	50.67
7	66.00	53.33	60.63	54.33	70.17	53.67
8	67.57	54.67	63.37	57.33	70.87	54.33
9	71.80	57.67	64.47	57.33	77.43	59.33
10	74.63	62.67	69.83	64.33	78.40	63.00
11	76.97	63.67	72.50	65.67	80.43	63.67
12	79.67	67.33	74.47	68.33	83.67	68.33
13	81.50	68.33	77.93	75.33	84.27	68.00
14	80.53	68.00	74.77	68.67	84.97	68.33
15	83.73	72.00	80.47	74.67	86.27	71.67
16	84.87	74.00	79.83	75.00	88.73	74.67
17	85.17	74.00	80.33	75.00	88.83	74.33
18	88.67	78.67	87.13	83.00	89.90	76.33
19	89.00	79.00	84.10	80.33	92.70	78.67
20	86.67	75.67	80.97	76.67	90.93	76.33
21	87.93	77.33	81.93	77.00	92.50	78.33
22	89.40	79.67	85.83	81.33	92.13	78.67
23	89.20	79.33	84.30	79.00	92.97	80.00
24	91.37	81.00	86.97	81.33	94.60	81.33
25	91.07	82.00	85.73	81.67	95.07	82.67
30	91.87	82.67	88.10	86.67	94.73	82.00
40	95.67	88.00	94.27	91.67	96.70	85.00
50	95.57	88.33	94.63	93.00	96.33	85.00
60	96.27	89.00	95.70	94.33	96.70	85.33
70	97.23	90.67	98.10	97.33	96.57	85.67
80	96.57	90.00	96.37	95.33	96.80	85.67
90	97.50	91.33	98.40	97.67	96.80	86.00
100	97.00	90.67	97.30	96.00	96.80	86.00
200	98.07	92.00	99.67	99.33	96.80	86.00
300	98.20	92.00	100.00	100.00	96.80	86.00
400	97.70	92.00	100.00	100.00	96.80	86.00
500	98.20	92.00	100.00	100.00	96.80	86.00
600	98.20	92.00	98.90	98.33	96.80	86.00
700	98.20	92.00	100.00	100.00	96.80	86.00
800	98.20	92.00	100.00	100.00	96.80	86.00
900	98.20	92.00	100.00	100.00	96.80	86.00
1024	98.20	92.00	100.00	100.00	96.80	86.00

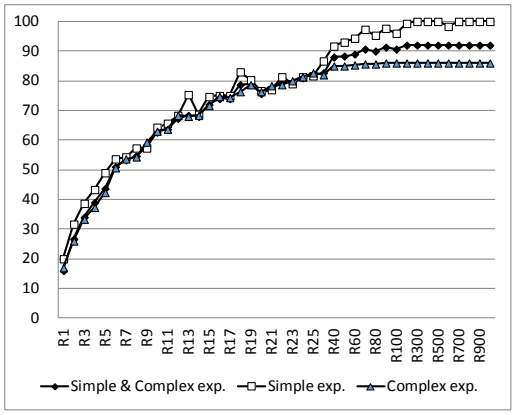

Fig. 9. MC/DC coverage by Testwell CTC++ for Random tests (20 variables).

- Maximal level of MC/DC coverage for complex expressions did not reach 100%.

However, numerical data were slightly different for 20 variables vs. 10 variables:

- For the simple expressions, MC/DC coverage reached 99% level for 200 random test cases and complete 100% MC/DC coverage was achieved started from 400 tests.
- For the complex expressions, the maximal possible levels of MC/DC coverage was 96.8% by CodeCover and 86% by CTC++.
- For the complex expressions, the maximal levels of MC/DC coverage were reached after 100 random test cases.

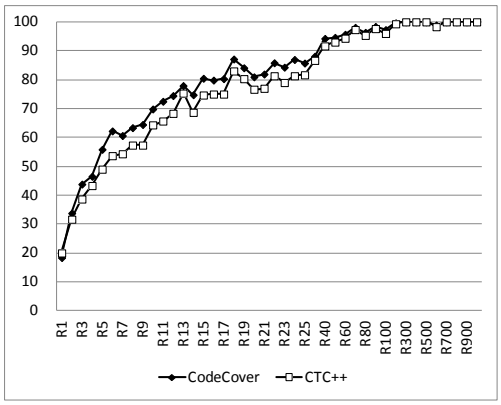

Fig. 10. Comparison of CodeCover and Testwell CTC++ results for Simple Expressions (20 variables).

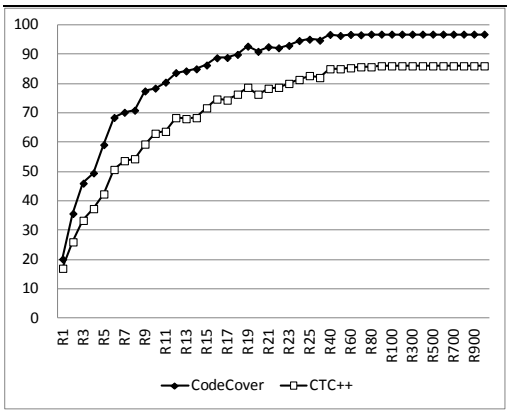

Fig. 11. Comparison of CodeCover and Testwell CTC++ results for Complex Expressions (20 variables).

In general, random test cases achieved a high level of MC/DC coverage very fast when the number of test increased. The precise data are given above but, very approximately, around 100 random tests provided high MC/DC coverage in all cases. Of course, this number is much higher than the amount of MC/DC test cases for one expression which is *n+1* for expressions size *n*, i.e., maximum 10 tests for expressions size 9. However, the number of different MC/DC tests necessary for all expressions together can be close to these 100 tests. At the same time, the process of MC/DC test generation is much harder comparing with random testing and should been done separately for each expression. It makes random testing a good basis for development new approaches to achieve MC/DC coverage.

V. COMBINATORIAL COVERAGE LEVEL OF RANDOM TESTING

In contrast to MC/DC, combinatorial coverage level does not depend on logical expressions in software and depends only on input variables.

In this section we evaluate *t*-way coverage of random test cases for *t*=2…6. Similar to Section IV, we consider this coverage for 10 input variables (Table VII) and 20 input variables (Table VIII). We use the same random test cases as in Section IV with sizes from 2 to 1024.

To understand how high the combinatorial coverage level of random test cases is, it is necessary to random sets of the same sizes as *t*-way sets. The sizes of combinatorial test sets for *t*=2…6 and for *n*=10 and *n*=20 are presented in Table VI. They are not optimal (minimal) but are close to optimal values and reflect sizes of combinatorial test sets generated by ACTS tool.

TABLE VI. SIZES OF COMBINATORIAL TEST SETS

Number of logical variables	2-way	3-way	4-way	5-way	6-way
10	10	20	44	93	178
20	12	27	66	165	375

TABLE VII. COMBINATORIAL COVERAGE FOR RANDOM TESTS (10 VARIABLES)

Random Test Set Size	2-way	3-way	4-way	5-way	6-way
2	46.85	24.51	12.44	6.25	3.12
3	60.55	34.27	18.08	9.25	4.67
4	68.33	41.24	22.63	11.86	6.08
5	76.85	49.51	28.12	14.96	7.70
6	83.52	55.49	32.11	17.31	9.01
7	85.74	59.23	35.48	19.58	10.33
8	85.92	63.96	39.61	22.20	11.78
9	92.78	70.48	44.58	25.28	13.47
10	96.85	76.70	49.29	27.94	14.86
11	94.44	76.60	51.57	30.10	16.20
12	98.52	81.39	54.71	32.17	17.52
13	97.78	83.40	57.68	34.31	18.76
14	97.78	83.16	58.00	35.03	19.48
15	99.26	87.95	63.13	38.37	21.18
16	98.15	86.84	63.43	39.53	22.30
17	99.44	91.42	69.30	43.66	24.43
18	99.44	90.73	68.26	43.27	24.63
19	99.81	94.10	73.06	46.64	26.40
20	98.89	91.18	70.29	45.43	26.21
21	99.81	94.69	75.81	50.24	29.09
22	99.81	95.69	77.41	51.41	29.87
23	100.00	95.49	77.80	52.35	30.76
24	100.00	96.84	79.70	54.14	32.02
25	100.00	97.51	81.58	56.31	33.43
30	100.00	98.23	86.07	61.76	37.85
40	100.00	99.51	92.74	72.51	47.37
50	100.00	99.96	97.12	81.48	55.84
60	100.00	100.00	96.32	84.80	60.76
70	100.00	100.00	99.01	93.29	75.71
80	100.00	100.00	99.53	93.01	73.16
90	100.00	100.00	99.65	94.43	76.68
100	100.00	100.00	99.87	96.60	81.34
200	100.00	100.00	100.00	99.96	97.42
300	100.00	100.00	100.00	100.00	99.61
400	100.00	100.00	100.00	100.00	99.98
500	100.00	100.00	100.00	100.00	100.00
600	100.00	100.00	100.00	100.00	100.00
700	100.00	100.00	100.00	100.00	100.00
800	100.00	100.00	100.00	100.00	100.00
900	100.00	100.00	100.00	100.00	100.00
1024	100.00	100.00	100.00	100.00	100.00

As it is possible to conclude from Tables VII and VIII, the level of combinatorial coverage of random test cases is quite high. Thus, in all situations for any *n* and *t*, this level for

random test sets of the same size as combinatorial test sets is around 90-97%.

The level of combinatorial coverage grows very fast when the number of random tests increases. However, after 90% the increase becomes significantly slow. To reach 100% of combinatorial coverage, significantly more random tests are required comparing with the combinatorial test sets. Thus, 23 (for n=10) and 24 (for n=20) random test cases are necessary to archive 100% pairwise coverage comparing with 10 and 12 test cases in pairwise test sets. The similar situations are for t-way coverage as it is possible to see from Tables VII and VIII.

TABLE VIII. COMBINATORIAL COVERAGE FOR RANDOM TESTS (20 VARIABLES)

Random Test Set Size	2-way	3-way	4-way	5-way	6-way
2	46.57	24.43	12.41	6.23	3.12
3	58.34	33.41	17.80	9.16	4.64
4	64.02	38.91	21.67	11.54	6.01
5	77.19	48.80	27.50	14.62	7.54
6	84.61	56.70	32.82	17.60	9.10
7	88.03	61.58	36.68	20.02	10.46
8	90.48	65.84	40.40	22.46	11.86
9	92.19	68.71	43.16	24.45	13.07
10	94.38	74.22	48.49	28.14	15.29
11	95.97	77.20	50.84	29.45	15.89
12	96.88	80.29	54.19	31.79	17.26
13	97.24	81.22	55.66	33.22	18.27
14	98.64	84.71	59.52	35.94	19.83
15	98.64	85.55	61.20	37.54	20.94
16	99.47	90.09	66.24	40.86	22.71
17	99.65	89.75	66.19	41.38	23.33
18	99.69	91.83	69.41	43.92	24.87
19	99.56	91.38	69.98	45.04	25.82
20	99.56	92.38	71.31	46.10	26.54
21	99.83	94.10	73.94	48.33	27.96
22	99.96	95.95	77.69	51.54	29.88
23	99.91	95.73	78.03	52.40	30.72
24	100.00	96.79	80.17	54.36	31.96
25	99.96	96.99	81.05	55.43	32.81
30	100.00	98.69	86.76	62.43	38.18
40	100.00	99.60	92.40	71.66	46.51
50	100.00	99.89	96.10	79.74	54.68
60	100.00	99.94	97.82	84.91	60.86
70	100.00	99.99	98.86	88.99	66.54
80	100.00	100.00	99.52	92.51	72.08
90	100.00	100.00	99.64	93.97	75.40
100	100.00	100.00	99.84	95.59	78.80
200	100.00	100.00	100.00	99.83	95.64
300	100.00	100.00	100.00	99.99	99.10
400	100.00	100.00	100.00	100.00	99.79
500	100.00	100.00	100.00	100.00	99.96
600	100.00	100.00	100.00	100.00	99.99
700	100.00	100.00	100.00	100.00	100.00
800	100.00	100.00	100.00	100.00	100.00
900	100.00	100.00	100.00	100.00	100.00
1024	100.00	100.00	100.00	100.00	100.00

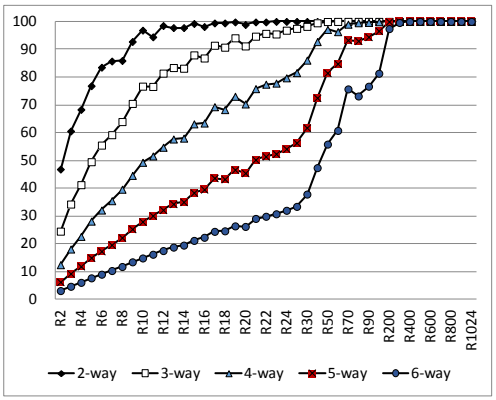

Fig. 12. Combinatorial coverage for Random tests (10 variables).

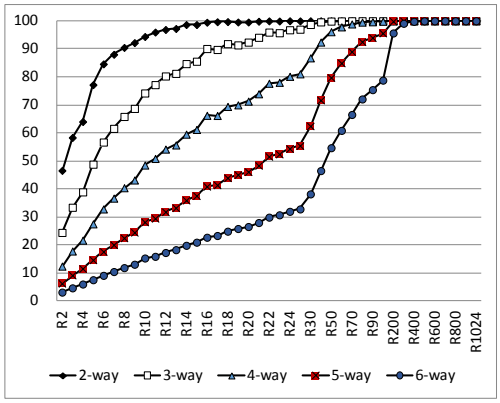

Fig. 13. Combinatorial coverage for Random tests (20 variables).

VI. CONCLUSIONS AND FUTURE WORK

This paper evaluates the ability of random testing to provide coverage according to MC/DC and t-way testing criteria. One hundred logical expressions of different sizes were generated. We also generated 252 random test sets with from 2 to 1024 test cases in a set. These sets were used to test a software program with logical expressions and the levels of coverage were evaluated using the structural coverage tools CodeCover, Testwell CTC++, and CCM, which measures the coverage of input value combinations in a test suite.

Our experiments show that for simple expressions, when the number of random test cases increased, they quickly reach a high level of coverage both for MC/DC and 2-way. The close to 100% level of MC/DC coverage is achieved after approximately 100 random tests in many cases, but full coverage may require doubling test set size. Complex expressions required a much larger test set size for MC/DC coverage on the order of 90%, again doubling the number of tests for full coverage. The paper provides detailed data for different types of expressions and different testing tools.

An unexpected result of the study was that structural coverage tools differ in their definition of partial MC/DC coverage, resulting in significant variation in coverage calculations. The primary standard for MC/DC coverage,

RTCA DO-178B, requires full coverage, so partial coverage numbers are generally not used. In cases where knowledge of less than complete MC/DC coverage is of interest, a consistent definition of partial coverage will need to be specified.

Random test sets of the same sizes as *t*-way sets provide the 90-97% level of combinatorial coverage. However, much more random tests are required to reach 100% coverage. The paper provides detailed data for different numbers of input variables, different types of expressions, and *t*-way coverage for *t*=2…6.

The obtained results can help for integration random testing with other approaches (in particular, MC/DC and combinatorial testing) to increase of effectiveness of testing software with complex logical structure.

ACKNOWLEDGMENT

This work was performed under the following financial assistance award 70NANB15H217 from the U.S. Department of Commerce, National Institute of Standards and Technology.

Disclaimer: *Products may be identified in this document, but identification does not imply recommendation or endorsement by NIST, nor that the products identified are necessarily the best available for the purpose.*

REFERENCES

[1] International Standard ISO/IEC/IEEE 29119-1:2013 "Software and systems engineering — Software testing — Part 1: Concepts and definitions," 2013.

[2] M. Grindal, J. Offutt, S. Andler, "Combination Testing Strategies: a Survey," Software Testing, Verification and Reliability, Vol. 15, No. 3, pp. 167-199, 2005.

[3] D. R. Kuhn, R. Kacker, and Y. Lei, Introduction to Combinatorial Testing, Chapman and Hall/CRC, 2013, 341 pages.

[4] D. R. Kuhn, R. Kacker, Y. Lei, and J. Hunter, "Combinatorial software testing, " IEEE Computer, vol. 42, no. 8, August 2009.

[5] RTCA/DO-178B, "Software Considerations in Airborne Systems and Equipment Certification," RTCA, Washington D.C., USA, 1992.

[6] J. Chilenski and S. Miller, "Applicability of Modified Condition/Decision Coverage to software testing," Software Engineering Journal, September 1994, pp. 193-200.

[7] RTCA/DO-178B "Software Considerations in Airborne Systems and Equipment Certification," Radio Technical Commission for Aeronautics, 1992.

[8] RTCA/DO-178C "Software Considerations in Airborne Systems and Equipment Certification," Radio Technical Commission for Aeronautics, 2012.

[9] Y. Moy, E. Ledinot, H. Delseny, V. Wiels, and B. Monate, "Testing or Formal Verification: DO-178C Alternatives and Industrial Experience," IEEE Software, May/June 2013, Volume 30, Issue 3, pp. 50-57.

[10] M. R. Girgis and M. R. Woodward, "An experimental comparison of the error exposing ability of program testing criteria," Proceedings of the Workshop on Software Testing, pp. 64-73. IEEE Computer Society Press, July 1986.

[11] V. Basili and R. Selby, "Comparing the Effectiveness of Software Testing Strategies," IEEE Trans. Softw. Eng. SE-13, (December 1987), pp. 1278–1296.

[12] C.-A. Sun, Y. Zai, and H. Liu, "Evaluating and comparing fault-based testing strategies for general boolean specifcations: A series of experiments," The Computer Journal, 2015, Volume 58, Issue 5, pp. 1199-1213.

[13] P.S. Kochhar, F. Thung, and D. Lo, "Code coverage and test suite effectiveness: Empirical study with real bugs in large systems," Proceedings of the IEEE 22nd International Conference on Software Analysis, Evolution and Reengineering (SANER), Montreal, Canada, 2-6 March 2015, pp. 560-564.

[14] S. Vilkomir and D. Anderson, "Relationship between pair-wise and MC/DC testing: Initial experimental results," Proceedings of the IEEE 8th International Conference on Software Testing, Verification and Validation Workshops (ICSTW 2015), 13-17 April 2015, Graz, Austria.

[15] P. Thevenod-Fosse, H. Waeselynck, and Y. Crouzet, "An experimental study on software structural testing: deterministic versus random input generation," Proceedings of the 21st International Symposium on Fault-Tolerant Computing (FTCS 91), IEEE Press, Jun. 1991, pp. 410-417.

[16] M. A. Vouk, K-C. Tai, and A. Paradkar, "Empirical studies of predicate-based software testing," Proceedings of the 5th International Symposium on Software Reliability Engineering, 1994.

[17] T. Y. Chen, F. C. Kuo, H. Liu, and W. E. Wong, "Code coverage of adaptive random testing," IEEE Transactions on Reliability, 2013, 62(1), pp. 226-237.

[18] S. Vilkomir, O. Starov, and R. Bhambroo, "Evaluation of t-way Approach for Testing Logical Expressions in Software," Proceedings of the IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops (ICSTW 2013), 18–20 March 2013, Luxembourg, pp. 249-256.

[19] W. Ballance, S. Vilkomir, and W. Jenkins, "Effectiveness of Pair-wise Testing for Software with Boolean Inputs," Proceedings of the Fifth International Conference on Software Testing, Verification and Validation (ICST 2012), April 17-21, 2012, Workshop on Combinatorial Testing (CT-2012), Montreal, Canada, pp. 580-585.

[20] S. Liu and Y. Chen, "A relation-based method combining functional and structural testing for test case generation," Journal of Systems and Software 81.2 (2008), pp. 234-248.

[21] C. Pfaller and M. Pister, "Combining Structural and Functional Test Case Generation," Proceedings of the Software Engineering Conference (SE08) , Munich, February 2008., pp. 229-241.

[22] C. D. Nguyen, A. Marchetto, and P. Tonella, "Combining model-based and combinatorial testing for effective test case generation," Proceedings of the 2012 International Symposium on Software Testing and Analysis, pp. 100-110. ACM, 2012.

[23] E. P. Enoiu, K. Doganay, M. Bohlin, D. Sundmark, and P. Pettersson, "MOS: an integrated model-based and search-based testing tool for function block diagrams," Proceedings of the 1st International Workshop on Combining Modelling and Search-Based Software Engineering, pp. 55-60. IEEE Press, 2013.

[24] R. Bartholomew, "An industry proof-of-concept demonstration of automated combinatorial test," Proceedings of the 8th International Workshop on Automation of Software Test (AST'13), May 18-19, 2013, San Francisco, CA, USA, pp. 118-124.

[25] "CodeCover," http://codecover.org

[26] R. Schmidberger, "Well-Defined Coverage Metrics for the Glass Box Test," In Testing Software and Systems, pp. 113-128. Springer Berlin Heidelberg, 2014.

[27] Testwell, "Testwell CTC++. Test Coverage Analyzer for C/C++," http://www.testwell.fi/ctcdesc.html

[28] Verifysoft Technology GmbH, "Testwell CTC++ Test Coverage Analyser," http://www.verifysoft.com/en_ctcpp.html

[29] National Institute of Standards and Technology (NIST), "Combinatorial Coverage Measurement Tool, User Guide, January 30, 2011," http://csrc.nist.gov/groups/SNS/acts/documents/ComCoverage110130.pdf

[30] I. D. Mendoza, D. R. Kuhn, R. N. Kacker, and Y. Lei, "CCM: A Tool for Measuring Combinatorial Coverage of System State Space," Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2013), 10-11 Oct. 2013, Baltimore, Maryland, USA, p. 291.

[31] J. Chilenski, "An investigation of three forms of the modified condition decision coverage (MCDC) criterion," Tech. Report DOT/FAA/AR-01/18, FAA, 2001.

[32] V. Durelli, et al., "What to expect of predicates: An empirical analysis of predicates in real world programs," Journal of Systems and Software 113, 2016, pp. 324-336.

[33] Kalimetrix, “Logiscope TestChecker,” http://www.kalimetrix.com/logiscope/testchecker

[34] Razorcat, “Automated testing of embedded software,” http://www.razorcat.eu/tessy.html